A Formal Characterization of Non-Monotonicity in Tensor Cores

Paul Jiang
jiang861@purdue.edu
Purdue University
West Lafayette, Indiana, United States

Vivian Zheng
vivian.zheng.1@stonybrook.edu
Stony Brook University
Stony Brook, New York, United States

Abstract

Modern high-performance computing increasingly relies on hardware accelerators like NVIDIA Tensor Cores, which employ non-standard internal arithmetic that can evolve between hardware generations. This non-standard approach can violate the fundamental mathematical property monotonicity, leading to incorrect outputs where adding a larger number produces a smaller result. To address this, we introduce a formal framework using Satisfiability Modulo Theories to analyze the hardware design space by systematically varying hardware features (e.g. number of terms (n), internal padding bits (p)) within a custom bitvector encoding. We derive a precise condition for guaranteed monotonicity, proving that non-monotonicity can only occur when $p \leq \lfloor \log_2(n-1) - 2 \rfloor$. We also derive a formula for the maximum magnitude of error when non-monotonicity can occur. Our results provide hardware architects with provably correct design parameters to eliminate such anomalies, ensuring greater numerical stability.

CCS Concepts

• **Theory of computation** → *Formalisms*; • **Theorem proving and SAT solving**; • **Computing methodologies**;

Keywords

Formal Methods, High-Performance Computing, Artificial Intelligence, Floating-Point Arithmetic, Formal Verification, Reproducibility, Matrix Accelerators

ACM Reference Format:
Paul Jiang and Vivian Zheng. 2025. A Formal Characterization of Non-Monotonicity in Tensor Cores. In *SC '25: Student Research Competition Poster Session, November 16–21, 2025, St. Louis, MO.* ACM, New York, NY, USA, 2 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Summary

The demand for computational power in fields like artificial intelligence and scientific computing has led to the widespread adoption of specialized hardware accelerators, most notably NVIDIA Tensor Cores [7]. These accelerators achieve their performance by using a simpler internal circuitry that deviates from the well-defined IEEE 754 standard for floating-point. This divergence creates a significant challenge for software portability and numerical reproducibility, as code that behaves predictably on one system may produce subtle, unexpected errors on another [4, 6]. A notable consequence of this non-standard arithmetic is the violation of fundamental mathematical properties that engineers and algorithms assume to hold. This research focuses on one such property: monotonicity. In standard arithmetic, monotonicity is simple and intuitive. If one increases an input to an addition, the sum should similarly increase or stay the same; it should never decrease. Formally, if $a \geq b$, then it should always be true that $x + a \geq x + b$. However, recent studies have demonstrated, this is in truth, not always the case within modern accelerators. Work by Fasi et al. has shown that due to specific design choices—namely, omitting normalization steps between intermediate additions in a multi-term sum, non-monotonicity can occur [3, 5]. This means that adding a larger number can produce a smaller result and is observable in many widely-used commercial hardware. For example, on the Volta GPU, the following behavior has been observed [3]:

$$2^0 + \sum_{i=1}^{4} 2^{-24} = 2^0$$

$$(2^0 - 2^{-24}) + \sum_{i=1}^{4} 2^{-24} = 2^0 + 2^{-23}$$

These observations were subsequently further studied by Mikaitis [5], who showed that the root cause of such non-monotonicity is due to a gradual precision growth, where an intermediate sum crossing a power-of-two boundary does not trigger a right-shift normalization. Instead, the adder uses extra padding bits (carry-out bits) at the top of the accumulator to handle the overflow. This allows the accumulator to represent the larger integer part of the sum without discarding the original, least significant bits of the mantissa. However, while Mikaitis explains the underlying mechanism, we work to formalize these behaviors to derive the exact hardware conditions and bounds that govern non-monotonicity.

In order to formally and exhaustively characterize this behavior across the entire hardware design space, we adopt a formal methods approach using Satisfiability Modulo Theories (SMT) solvers [1, 2]. Our work builds upon that of Valpey et al., who previously formalized the internal arithmetic of three generations of NVIDIA Tensor Cores (Volta, Turing, and Ampere) [8]. Their work provided executable specifications of these hardwares' behaviors, such as the exact rounding mode and lack of intermediate normalization, creating a verified basis for our work.

Leveraging these insights, we construct a parametric SMT model to systematically explore the relationship between hardware design choices and numerical properties. Our automated framework varies key hardware parameters, namely the number of terms being added (n) and the number of internal padding bits (p) used to handle precision loss from the accumulator. These parameters are most

SC '25, Nov 16-21, 2025, St. Louis, MO

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

relevant as they directly govern the mechanism of precision growth and overflow.

For each (n, p) configuration, we generate a logical query challenging the SMT solver to find a counterexample that violates monotonicity. By iteratively incrementing p for each n until the solver can prove that no counterexample exists (returning UNSAT), we precisely identify the boundary at which monotonicity is guaranteed. From this aggregated data, we are able to manually derive a general rule that characterizes this boundary. We verify our rule, by feeding it back into the solver as a new constraint to our SMT query and challenging the solver to find a contradiction. An UNSAT result from this challenge formally proves that our rule is exhaustive and correct for the entire problem space.

This formal process culminates in two key findings. First, we give a provably correct rule for guaranteeing monotonicity. We demonstrate that non-monotonicity can only occur when the number of padding bits p is less than or equal to the derived threshold.

$$p \leq \lfloor \log_2(n-1) - 2 \rfloor \tag{1}$$

Second, for configurations where non-monotonicity is possible, we derive a formula to calculate the maximum magnitude of the resulting error (M). For FP32 accumulation, this is given by:

$$M(n, p, m_{exp}) = 2^{(m_{exp} - (24 + p - \log_2(n-1)))} - 2^{(m_{exp} - 23)} \tag{2}$$

where m_{exp} is the maximum exponent of the addends.

The consequences of these findings are twofold. For hardware architects, Equation 1 provides a direct, verifiable method to design future accelerators that are provably free from this numerical anomaly. For software developers and numerical analysts working with existing hardware, Equation 2 is a critical tool to bound potential error, enabling the development of more robust algorithms and error-correction schemes. We provide a direct path toward designing more reliable and numerically stable hardware, ensuring that future high-performance computing applications can safely leverage accelerators.

Acknowledgments

We would like to express our sincere gratitude to Ganesh Gopalakrishnan (Faculty Advisor at University of Utah) for his many hours of advice. This research was supported under REU Site NSF 2244492 Trust and Reproducibility Education for Undergraduates. The authors did this work during their 10-week stay during the summer of 2025 at the University of Utah. We also acknowledge the facilities offered by the Kahlert School of Computing.

References

[1] Haniel Barbosa, Clark Barrett, Martin Brain, Gereon Kremer, Hanna Lachnitt, Makai Mann, Abdalrhman Mohamed, Mudathir Mohamed, Aina Niemetz, Andres Nötzli, Alex Ozdemir, Mathias Preiner, Andrew Reynolds, Ying Sheng, Cesare Tinelli, and Yoni Zohar. 2022. cvc5: A Versatile and Industrial-Strength SMT Solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, Dana Fisman and Grigore Rosu (Eds.). Springer International Publishing, Cham, 415–442.

[2] Leonardo De Moura and Nikolaj Bjørner. 2008. Z3: an efficient SMT solver. In *Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems* (Budapest, Hungary) *(TACAS'08/ETAPS'08)*. Springer-Verlag, Berlin, Heidelberg, 337–340.

[3] Massimiliano Fasi, Nicholas J Higham, Mantas Mikaitis, and Srikara Pranesh. 2021. Numerical behavior of NVIDIA tensor cores. *PeerJ Computer Science* 7 (10 Feb 2021), e330. https://doi.org/10.7717/peerj-cs.330

[4] Xinyi Li, Ang Li, Bo Fang, Katarzyna Swirydowicz, Ignacio Laguna, and Ganesh Gopalakrishnan. 2024. FTTN: Feature-Targeted Testing for Numerical Properties of NVIDIA AMD Matrix Accelerators. arXiv:2403.00232 [cs.AR] https://arxiv.org/abs/2403.00232

[5] Mantas Mikaitis. 2023. Monotonicity of Multi-Term Floating-Point Adders. arXiv:2304.01407 [cs.MS] https://arxiv.org/abs/2304.01407

[6] Fraser Mince, Dzung Dinh, Jonas Kgomo, Neil Thompson, and Sara Hooker. 2023. The Grand Illusion: The Myth of Software Portability and Implications for ML Progress. arXiv:2309.07181 [cs.SE] https://arxiv.org/abs/2309.07181

[7] Daniel Reed, Dennis Gannon, and Jack Dongarra. 2023. HPC Forecast: Cloudy and Uncertain. *Commun. ACM* 66, 2 (jan 2023), 82–90. https://doi.org/10.1145/3552309

[8] Benjamin Valpey, Xinyi Li, Sreepathi Pai, and Ganesh Gopalakrishnan. 2025. An SMT Formalization of Mixed-Precision Matrix Multiplication: Modeling Three Generations of Tensor Cores. arXiv:2502.15999 [cs.AR] https://arxiv.org/abs/2502.15999